LA-UR-14-23598
Approved for public release; distribution is unlimited.

Title: Computational Co-Design of a Multiscale Plasma Application: A Process and Initial Results

Author(s): Payne, Joshua E.
Knoll, Dana A.
McPherson, Allen L.
Taitano, William
Chacon, Luis
Chen, Guangye
Pakin, Scott D.

Intended for: 28th IEEE International Parallel & Distributed Processing Symposium, 2014-05-19/2014-05-23 (Phoenix, Arizona, United States)

Issued: 2014-05-21

Computational Co-Design of a Multiscale Plasma Application
A Process and Initial Results

Joshua Payne, Dana Knoll, Allen McPherson, William Taitano, Luis Chacón, Guangye Chen, and Scott Pakin

Los Alamos National Laboratory

May 22, 2014

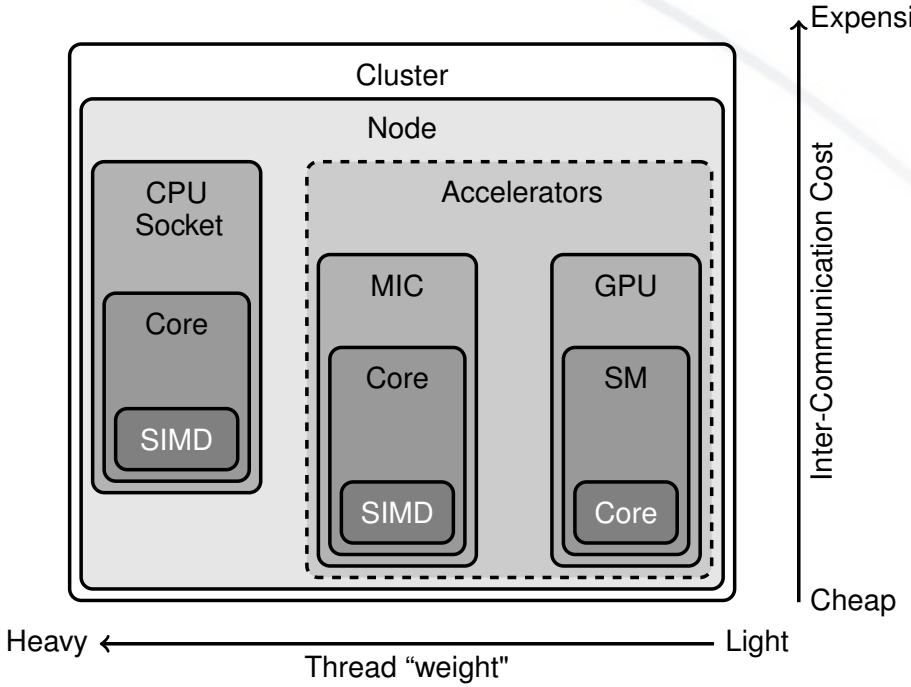
Presented by Joshua Payne

Outline

Trending Toward Hierarchical Architectures

Emerging architectures introduce:

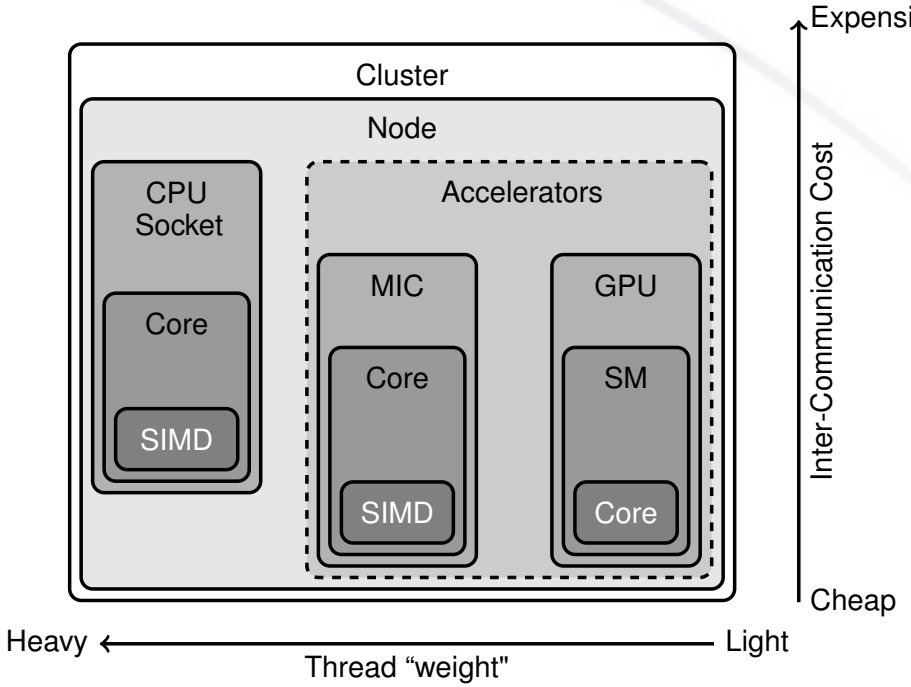
- Levels of parallelism.
- Heterogeneity.
- Parallel Branches.
- Complex memory systems.
- FLOPS over Bandwidth.



Trending Toward Hierarchical Architectures

Emerging architectures introduce:

- Levels of parallelism.
- Heterogeneity.
- Parallel Branches.
- Complex memory systems.
- FLOPS over Bandwidth.



Current algorithms and applications do not map well to these new architectures!

Algorithms, Abstractions, and Applications

Los Alamos NATIONAL LABORATORY EST. 1943

New Algorithms must:

- Utilize very fine-grained parallelism.
- Isolate a large amount of work on accelerators.
- Minimize communication between host and accelerators.
- Isolate work on-node.
- Maximize use of SIMD and SIMT features.

New Applications must:

- Exhibit properties similar to emerging architecture.
- Achieve performance and portability across architectures.
- Achieve a significant new physics simulation capability.
- Provide feedback to architecture development.

Algorithms, Abstractions, and Applications

Los Alamos NATIONAL LABORATORY EST. 1943

New Algorithms must:

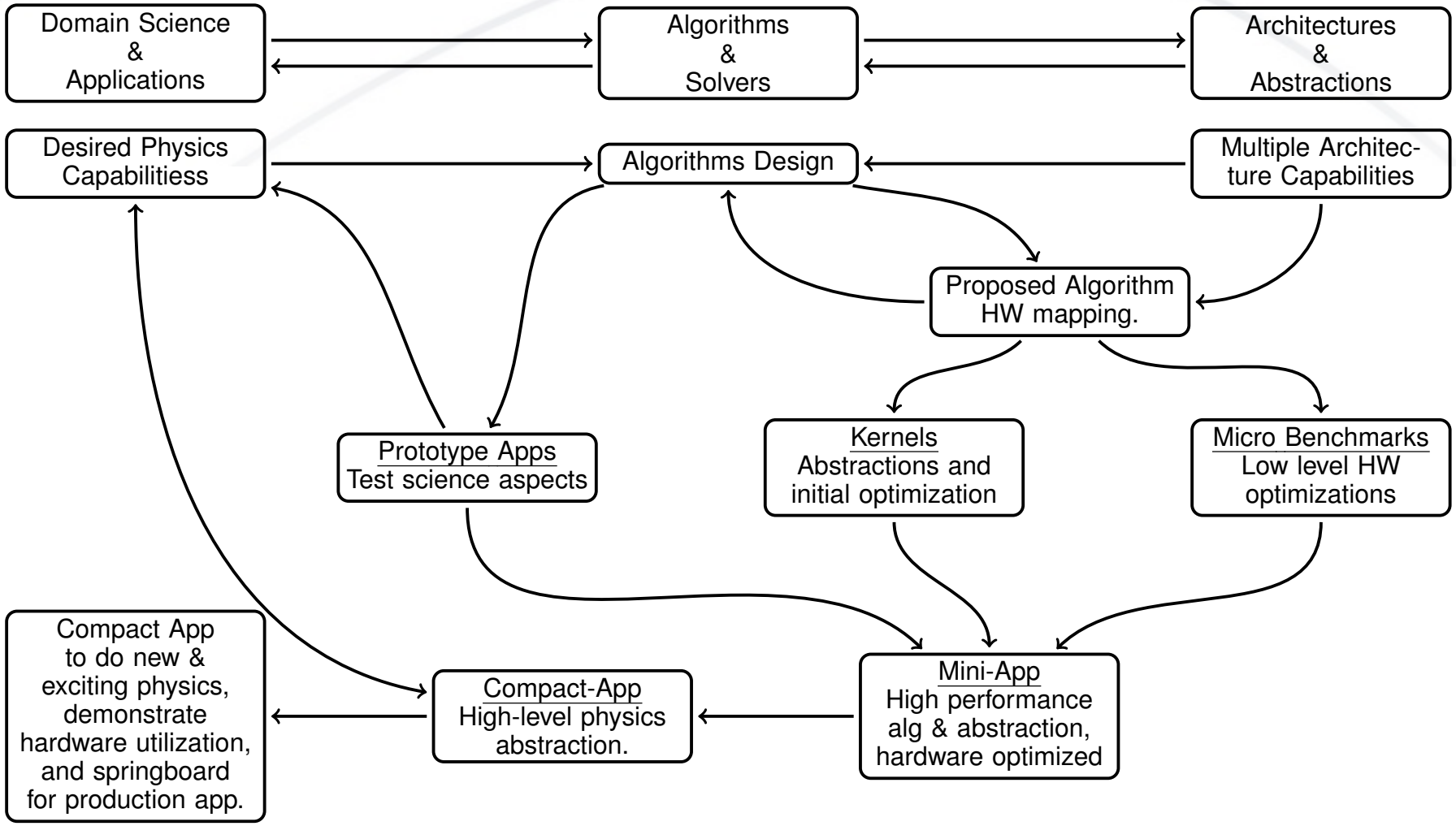
- Utilize very fine-grained parallelism.
- Isolate a large amount of work on accelerators.
- Minimize communication between host and accelerators.
- Isolate work on-node.
- Maximize use of SIMD and SIMT features.

New Applications must:

- Exhibit properties similar to emerging architecture.
- Achieve performance and portability across architectures.
- Achieve a significant new physics simulation capability.
- Provide feedback to architecture development.

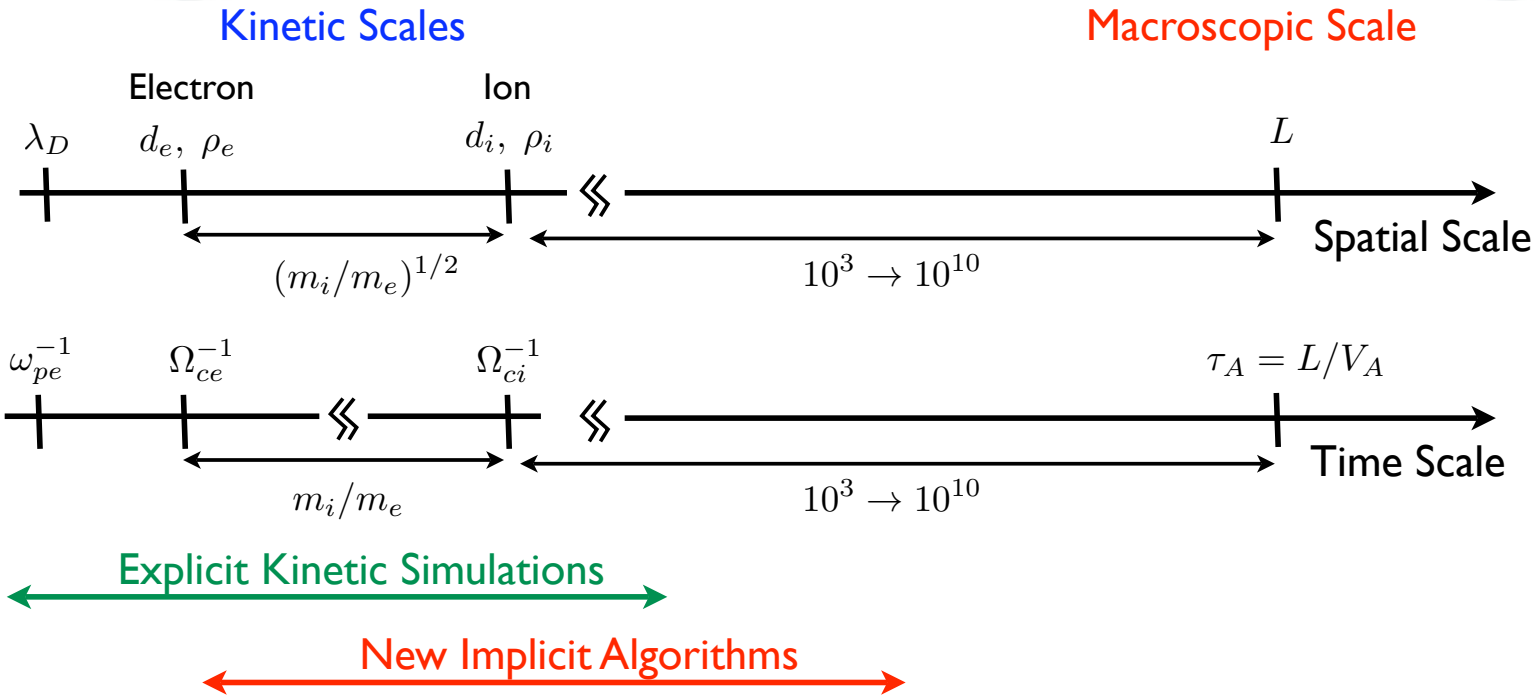
We propose an evolving co-design process for developing applications and algorithms with these characteristics.

Evolving Co-Design Process



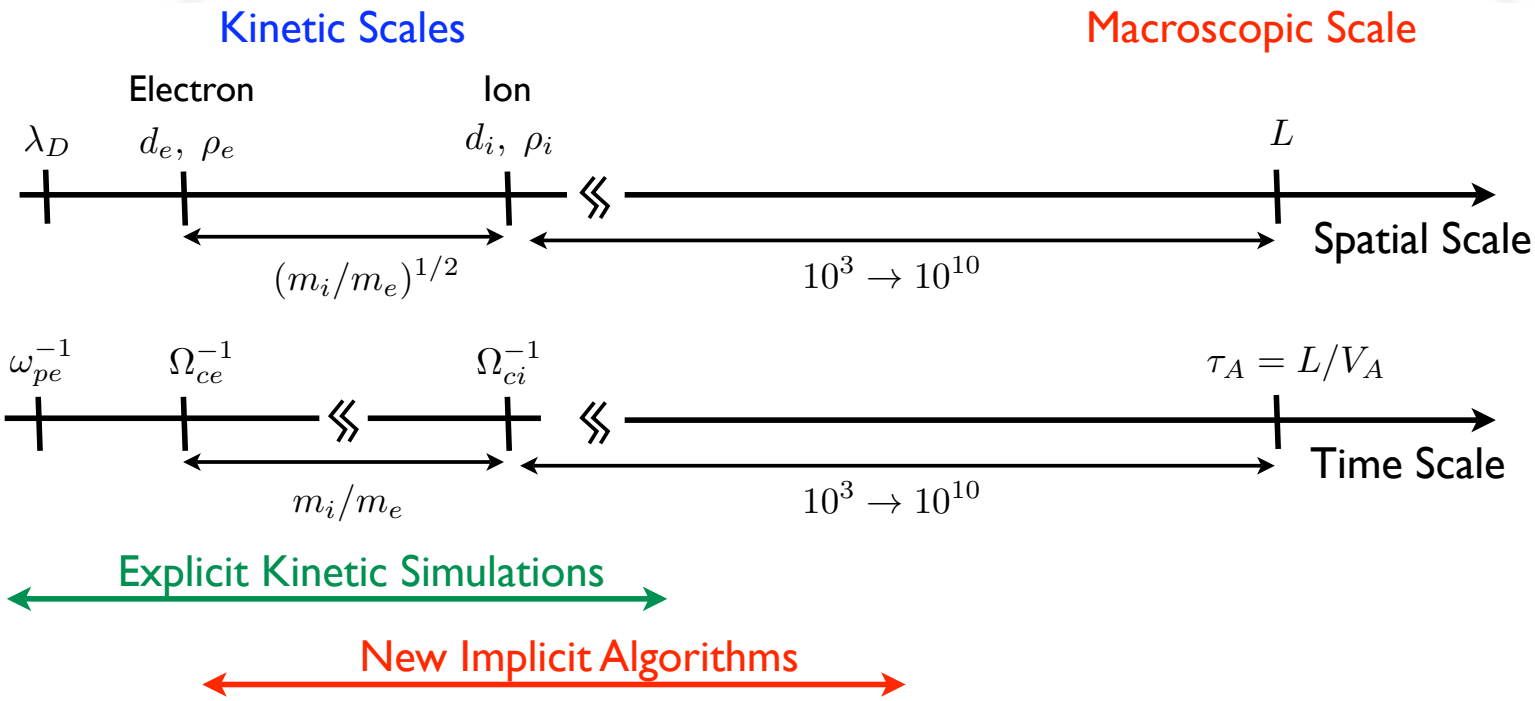
Domain Science and Applications

- Multi-Scale physics problems exhibit a hierarchical nature.
- Kinetic plasma simulations are a prime example.



Domain Science and Applications

- Multi-Scale physics problems exhibit a hierarchical nature.
- Kinetic plasma simulations are a prime example.



We need algorithms that allow for the isolation of scales at various levels of parallelism.

Scale Bridging Algorithms

Particle Methods:

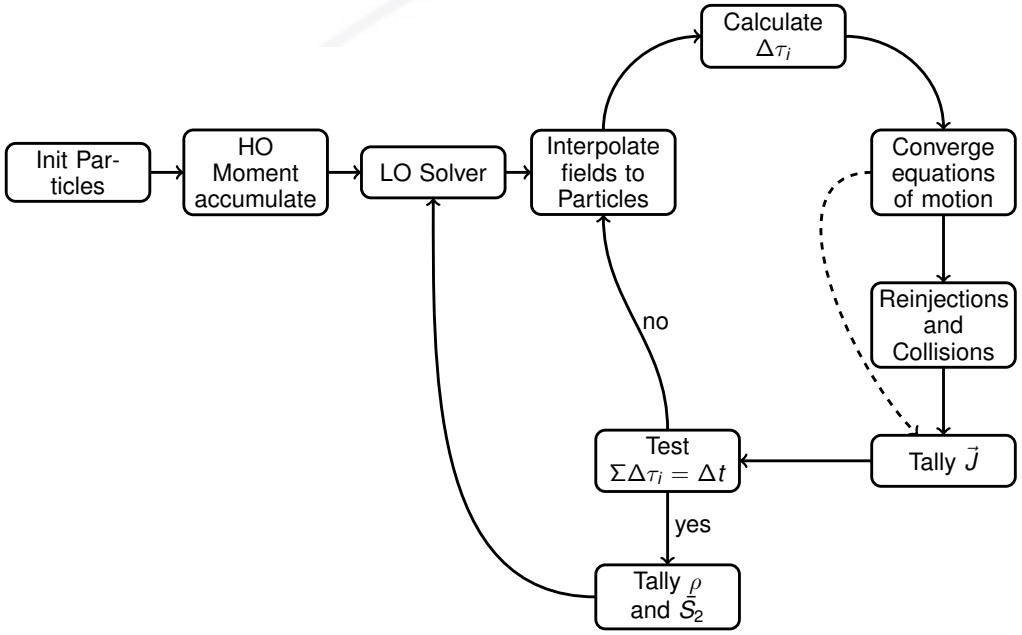
- Good:
  - High degree of isolated parallelism.
  - Maps well to accelerators.
  - Highly scalable.
  - Can resolve very small nonlinear features.
- Bad:
  - Frequent global communication required.
  - Small timesteps and fine meshes required.

Grid Methods:

- Good:
  - Resolves large-scale nonlinear features.
  - Can use coarser meshes.
  - Larger time steps.
- Bad:
  - Does not resolve small nonlinear effects.
  - Does not fit well on accelerators.

Solution: Use High-Order / Low-Order acceleration bridge these two methods.

Consistent Moment-Based Implicit PIC



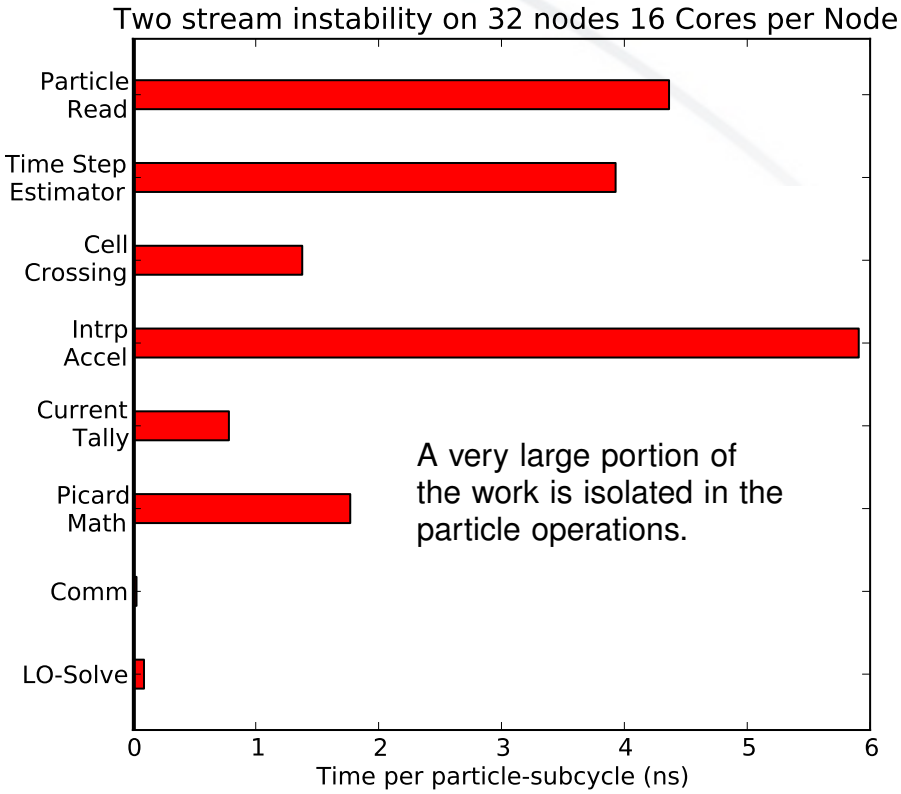
Moment-Based Implicit PIC with subcycling

for $t = 0 \rightarrow t_{max}$ **do**
  while resid $>$ tol **do** // Outer Picard Loop
    for all $p_{i,0} \in \mathcal{P}_t$ **do** // Particle Loop
      while $\sum \Delta\tau_\nu < \Delta t$ **do** // Subcycle Loop
        Interpolate $\vec{E}_{t+1/2}, \vec{B}_{t+1/2}$ to $p_{i,\nu}$
        Estimate $\Delta\tau_\nu$
        Solve equations of motions: $p_{i,\nu} \rightarrow p_{i,\nu+1}$
        Tally $\vec{J}(p_{i,\nu+1/2})$
      end while
      Tally $\rho(p_{i,t+1})$, $\bar{S}(p_{i,t+1})$
    end for
    Calculate residual
    $\vec{E}_{t+1}, \vec{B}_{t+1}$=LO($\vec{J}(p_{i,t+1/2})$, $\rho(p_{i,t+1})$, $\bar{S}(p_{i,t+1})$)
  end while
end for

Consistent Moment-Based Implicit PIC

Advantages:

- Charge conservation to machine precision.
- Energy conservation to specified tolerance.
- Significant amount of work isolated on node / accelerator.
- Majority of the work maps well to accelerators.



CoCoPIC Framework design

In order to support multiple mini-apps utilizing multiple optimization strategies we developed a flexible parallel implicit PIC framework called CoCoPIC.

The following design goals were incorporated into the development of CoCoPIC:

- Support multiple plasma physics problems of varying complexity, 1D1V to 2D3V.
- Support GPU, multi-core, and multi-node systems
- Develop abstraction layers to hide device specific optimizations from the physics code.
- Support multiple LO solvers, including TRILINOS based solvers[1]

[1] M. A. Heroux et. al. An overview of the trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.

Abstraction Layers

Science Abstractions accommodate:

- Different PDEs in Low Order system.
- Different Spatial and Velocity dimensionalities.
- Different normalization systems.

Hardware Abstractions accommodate:

- Different parallelization granularities.
- Different intrinsic operations
- Different data management capabilities
- Fast optimization testing

Proper development of abstraction layers can yield incredibly powerful code that is flexible, readable and high-performing.

PDE Abstraction Library

Given two ways to write the following PDE evaluated on a Yee mesh:

$$\frac{dp_x}{dt} + \frac{dS_{xx}n}{dx} + \frac{dS_{xy}n}{dy} - \frac{q}{m}nE_x = 0 \quad (1)$$

Hard-coded discretized form of our PDE:

```
res_px[i,j] = (px[t+1,i,j] - px[t,i,j])/dt
+ (Sxx[t+1/2,i,j]*n[t+1/2,i,j] - Sxx[t+1/2,i-1,j]*n[t+1/2,i-1,j])/dx
+ 0.25*(Sxy[t+1/2,i,j+1]*n[t+1/2,i,j+1] - Sxy[t+1/2,i,j-1]*n[t+1/2,i,j-1])/dy
+ 0.25*(Sxy[t+1/2,i-1,j+1]*n[t+1/2,i-1,j+1] - Sxy[t+1/2,i-1,j-1]*n[t+1/2,i-1,j-1])/dy
- q/m * Ex[t+1/2,i,j] * 0.5*(n[t+1/2,i,j]+n[t+1/2,i-1,j]) ;
```

Condensed form achieved through template meta-programming.

```
res_px(i,j) = (Dt(px) + Dx(Sxx*n) + IntrpX(IntrpY(Dy(Sxy*n)))
- q/m*Ex*IntrpX(n))(t+1/2,i,j);
```

The advantages of option 2 are as follows:

- Readability
- Type-safe. Each variable has a type based on its position on the mesh.
- Performance.

Code that looks like the math can also be fast!

Maintain single piece of code for expressing hi-order physics

- The high order physics does not change from device to device, only underlying operations and data management strategies.
- GPUs basically want one particle per thread.
- CPUs want many particles per thread with each operation vectorized.

For evaluating $c = a + b$, where a, b, and$c$ are all arrays. The GPU and CPU require a slightly different expression:

CPU Expression:

```
for(i=0;i<n;i+=4)
c[i] = vec_add(a[i], b[i]);
```

GPU Expression:

```
i = threadIdx.x
c[i] = a[i] + b[i];
```

With some use of template meta-programming we can hide both of these statements behind a statement such as `c = a + b`.

Analysis Tools

BYFL a tool developed at LANL by Scott Pakin and Pat McCormick that can be used to measure various performance aspects of a code[2] :

Metric	xRAGE	CoCoPIC
Ops per load	4.0063	10.9560
Flops per branch	0.7703	2.6791
Ops per branch	6.8952	26.9296
Bytes per flop	20.7993	15.7598
Bytes per op	2.3237	1.5679
Unique bytes per flop	0.1036	0.0001
Unique bytes per op	0.0116	0.0000
Bytes per unique byte	200.6784	144,986.9800

In-Situ Timers Built in high-resolution CPU timers that can be used to profile the code in-situ.

[2] S. Pakin. Compiler-based application analysis. *Exascale Research Conference, April 2012.*

Optimization Example: 2D3V CPU optimization

Optimization Strategy:

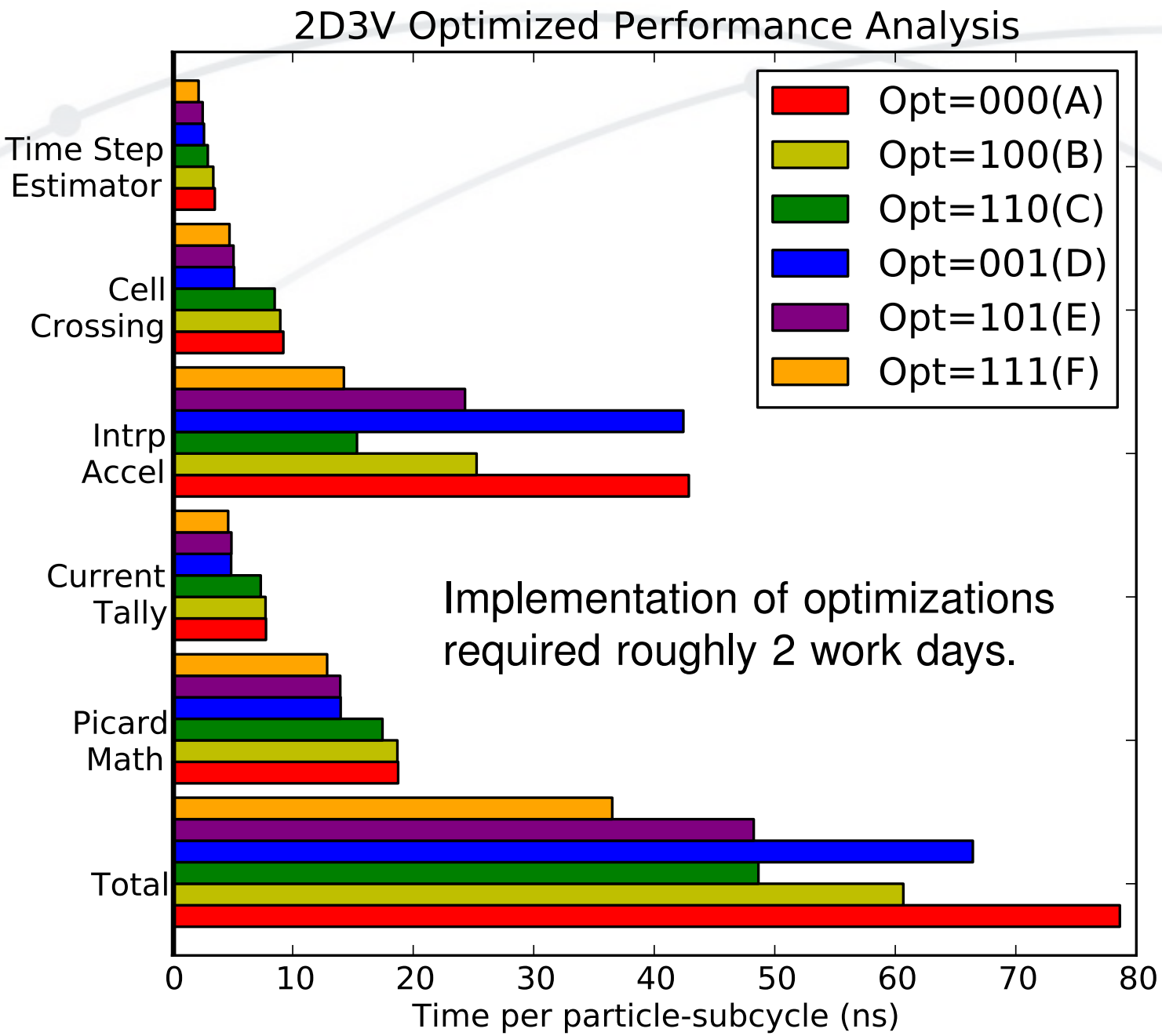
- Identify high runtime cost areas.
- Identify cause of high runtime cost (lots of flops, memory access patterns, etc.)
- Identify strategies to mitigate the causes of the high runtime costs.

Baseline - Performance results without extensive optimization.

Reduce FLOPs - Consolidates shape function evaluations.

Memory Access Patterns - Store the fields as an array of structures.

Hardware Intrinsics - Include hand vectorized routines.


2D3V Optimized Performance Analysis
Time Step Estimator
Cell Crossing
Intrp Accel
Current Tally
Picard Math
Total
Opt=000(A)
Opt=100(B)
Opt=110(C)
Opt=001(D)
Opt=101(E)
Opt=111(F)
Implementation of optimizations required roughly 2 work days.
0
10
20
30
40
50
60
70
80
Time per particle-subcycle (ns)

Conclusions

- Quality implementation of hierarchical algorithms (HO-LO) has clearly paved the way for maximizing on-node flops while minimizing node-to-node communication on a multi-node, many-core + GPU, architecture
- Readability and Performance win-win through high-level science abstractions.
- Portability, Readability, and Performance triple-win through hardware abstractions.
- **Achieved 20% - 40% of peak theoretical multi-core performance**
- More information in:

J. Payne, D. Knoll, A. McPherson, W. Taitano, L. Chacon, G. Chen, and S. Pakin. Computational co-design of a multiscale plasma application: A process and initial results. In *28th IEEE International Parallel & Distributed Processing Symposium (IEEE IPDPS 2014)*, Phoenix, USA, May 2014. (accepted).